Certification Authorities Software Team (CAST)

Position Paper
CAST-32

Multi-core Processors

COMPLETED May 2014 (Rev 0)

NOTE: **This position paper has been coordinated among representatives from certification authorities in North and South America, Europe, and Asia. However, it does not constitute official policy or guidance from any of the authorities. This document is provided for educational and informational purposes only and should be discussed with the appropriate certification authority when considering for actual projects.**

Table of Contents

NOTE: **This position paper has been coordinated among representatives from certification authorities in North and South America, Europe, and Asia. However, it does not constitute official policy or guidance from any of the authorities. This document is provided for educational and informational purposes only and should be discussed with the appropriate certification authority when considering for actual projects.**

NOTE: **This position paper has been coordinated among representatives from certification authorities in North and South America, Europe, and Asia. However, <u>it does not constitute official policy or guidance from any of the authorities</u>. This document is provided for educational and informational purposes only and should be discussed with the appropriate certification authority when considering for actual projects.**

Multi-Core Processors

1. PURPOSE

The purpose of this CAST paper is to identify topics with Multi-Core Processors (MCP) with two active cores that could impact the safety, performance and integrity of the software for a single airborne system executing on MCPs. For each topic, the paper provides rationale why these topics are of concern and objectives and suggested activities for the demonstration of compliance to CFR 25.1309(b) and CS-25.1309(a)(2).

2. BACKGROUND

MCPs are processors that contain two or more processing cores. Some manufacturers identify three classes of processors: single-core, dual-core and multi-core.

Even with two cores, MCPs are highly-complex devices that contain many additional features not present in single-core processors. The proposed use of MCPs in safety-critical airborne systems is a cause of concern to certification authorities for the following reasons:

- Some MCP additional features could cause interference between the applications executing simultaneously on the separate cores of an MCP. This interference has actually been observed during testing. Examples of these additional features are shared access to cache or other memory areas, operating systems / supervisors / hypervisors that can control and affect all the applications executing on all the cores, and 'coherency fabrics / coherency modules / interconnects' that control all the data transfers between the MCP cores, memory and the peripheral devices of the MCP via a shared bus.

- Many of these features that introduce interference channels between cores are built into the MCPs and were not designed or verified for compliance with the current airborne software or hardware guidance material. It may therefore be difficult or even impossible to fully characterize and verify all the possible effects of these features, which may include unintended and unexpected behavior. This leads to concerns that these features could cause a loss of integrity, a loss of availability or non-deterministic behavior of hosted applications due to such effects as variations in data access times, denial of access to data or to peripherals or by providing interference channels between applications. If safety-critical applications are to successfully execute on MCPs, the allowable data latency of each input parameter to an application may have to be analyzed so it is ensured that the applications can cope with the worst case variations in data access times, which should be measured. The overall execution times of applications may have to include allowances for such variations.

- MCPs contain many peripheral devices in addition to the processing cores. Many aspects of the MCPs, their peripheral devices and their debug functionality are configurable via a huge number of registers or pins, but it appears that some manufacturers do not provide complete public data on these configurable aspects and some may not provide complete lists of errata. Such undocumented and untestable features are potential sources of unintended functionality.

- Small variations in the known configuration settings can produce significant and unexpected variations in the behavior of an MCP, its cores and their applications and its peripheral devices. Careful consideration of all the possible settings and the preservation of those settings during operation are essential, even in the presence of problems such as single event upsets.

- Some MCPs contain functionality that could produce non-deterministic behavior due to functions that can dynamically reallocate processes or memory, dynamically activate / deactivate individual cores or dynamically alter their operating frequencies for reasons such as to save energy or to allocate processes to underused cores.

- Problems such as the greater complexity of MCPs, their features, the errata and configurable aspects that may not be fully documented and the fact that MCPs have not been previously used in safety-critical applications make the use of service history for MCPs unlikely to be acceptable as part of the justification for MCP installations.

- With MCPs, even if only one application executes on each core, those applications will require guaranteed access to a certain allocation of the memory, cache, databuses and peripheral devices of the entire MCP. The resources of the whole MCP will have to be managed and allocated to meet the demands of the applications executing on the separate cores.

- The hardware and built-in features of MCPs are not the only causes for concern, as the software that is hosted upon MCPs also requires careful consideration. In some software architectures, a single operating system (OS) may control all the applications hosted on all the cores. That OS may divide each software function into threads that execute in parallel on separate cores, and may then dynamically re-allocate the threads to different cores. These operating systems may not be compliant with ED-12C / DO-

178C and some may not be capable of providing partitioning between applications or deterministic execution of the applications.

3. REFERENCES AND RELATED DOCUMENTS / GUIDANCE AND STANDARDS

a. SAE ARP 4754a / EUROCAE ED 79a, Guidelines for Development of Civil Aircraft and Systems

b. AC 20-174, Development of Civil Aircraft and Systems

c. SAE ARP 4761 / EUROCAE ED 135, Guidelines and Methods for Conducting the Safety Assessment Process on Civil Airborne Systems and Equipment

d. RTCA DO-178B / EUROCAE ED 12B, Software Considerations in Airborne Systems and Equipment Certification

e. RTCA DO-178C / EUROCAE ED 12C, Software Considerations in Airborne Systems and Equipment Certification

f. AC 20-115C – Airborne Software Assurance

g. AMC 20-115C - Software Considerations for Certification of Airborne Systems and Equipment

h. EASA Certification Memorandum CM - SWCEH – 002, Software Aspects of Certification

i. RTCA DO-254 / EUROCAE ED-80, Design Assurance Guidance for AEH

j. Advisory Circular (AC) 20-152 / RTCA DO-254, Design Assurance Guidance for AEH

k. EASA Certification Memorandum CM - SWCEH – 001, Development Assurance of Airborne Electronic Hardware

4. DEFINITIONS

Applicable Software Guidance: the version of ED-12 / DO-178 that applies to the project, plus any certification authority software guidance (e.g. EASA Certification Review Item (CRI), AC 20-115C, etc.) applicable to the project.

Asymmetric Multi-processing (AMP): each individual functional process is permanently allocated to a separate core and each core has its own operating system. However, the operating systems may be multiple copies of the same operating system or be different from core to core.

Bound Multi-processing (BMP): extends symmetric multi-processing (SMP) by allowing the developer to bind any process and all of its associated threads to a specific core while using a common operating system across all cores.

Determinism / deterministic: The ability to produce a predictable outcome generally based on the preceding operations and data. The outcome occurs in a specified period of time with repeatability. (Ref DO-297/ED-124)

Multi-core processor (MCP): a device that contains two or more independent processing cores. A core in the MCP is defined as a device that executes software. This includes virtual cores (e.g. Intel's Hyperthreading microarchitecture).

Safety Critical: involving hosted software of DAL A, B or C. (Basis of definition – DO-254/ED-80 and DO-178B/ED-12B or DO-178C/ED-12C levels A, B and C correspond to failure conditions that respectively 'would prevent continued safe flight and landing', cause 'a large reduction in safety margins or functional capabilities.. potentially fatal injuries to a small number of those occupants' and 'a significant reduction in safety margins or functional capabilities.. possibly including injuries'.)

Safety Net: A safety net is defined as the employment of mitigations and protections at the appropriate level of aircraft and system design to help ensure continuous safe flight and landing. The safety net methodology focuses on the assumption that a microprocessor will misbehave. The ability to protect against unexpected behavior, damage, injury, and instability over the service life outside, or at a level above the device itself, is necessary as appropriate for the design assurance level. The safety net approach is a means to mitigate the risks associated with COTS microprocessors via both passive and active methods designed into aircraft systems. If it is not feasible to show that complex aircraft systems are sufficiently free of anomalous behavior by evaluating system components and system design, the safety net approach can mitigate unforeseen or undesirable COTS microprocessor operation by detecting and recovering from anomalous behavior at the operational system level. This approach requires the safety net to be designed as a function within the aircraft system. The safety net can include passive monitoring functions, active fault avoidance functions, and control functions for recovery of system operations. System architecture and control and recovery functions should be designed to facilitate effective system recovery from anomalous events. Safety nets should show that systems are sufficiently impervious to anomalous behavior by ensuring continuous functional availability and reliability, satisfying applicable regulations, and meeting airworthiness requirements. This includes verifying any disabled functionality from the COTS will remain inactive in the specific application. (Ref DOT/FAA/AR-11/5, Microprocessor Evaluations for Safety-Critical, Real-Time Applications, May 2011.)

Symmetric Multi-processing (SMP): a single operating system controls the execution of the processes on all the cores and may dynamically allocate sections of processes to run in parallel on separate cores.

5. POSITION

a. Objectives

All the objectives of this paper apply when the highest DAL of any of the software applications hosted by the MCP is DAL A or B. Some objectives are not applicable when the highest DAL of any of the applications hosted is DAL C. The objectives that apply according to the assigned DAL (A, B or C) of the hosted software are shown in section 6.k. of this document. Some of the objectives may not apply to a specific MCP implementation.

b. Activities

For each topic in this paper, suggested activities are documented in the Appendix at the end of the paper and are considered adequate and sufficient to comply with the objectives for each topic. These suggested set of activities are not mandatory. They are provided since the use of MCPs in civil aerospace is new and there is no industry guidance. The Certification Authority would review the applicant's proposed activities for acceptability.

An applicant should:

- state in their PHAC and PSAC or other deliverable document which activities they intend to conduct in order to fully comply with each of the objectives that are applicable to their installation,
- conduct those intended activities, and
- document the evidence from the activities that shows compliance with each objective for possible inspection by the certification authority.

c. More than Two Active Cores

The paper has not yet been extended for MCPs with more than two active cores.

d. Single Systems Applicability

Additional considerations beyond what is documented in this paper may be required for MCPs used in integrated modular avionics (IMA) applications. Considerations for IMAs that may need to be addressed include inter-system safety analyses, incremental certification, robust partitioning between applications of different systems, allocation of applications to processors, and error detection and handling to ensure that the loss or erroneous behavior of an application from one system could not affect a safety-critical application from another system that was hosted on the same MCP. The Certification Authorities are not currently aware of any MCP hardware and software implementations that would allow applications from more than one system to be partitioned in time on an MCP in the way that time partitioning is currently ensured for the applications of an IMA on a single core processor (SCP). Therefore, this current paper is limited to applications of one system to contain the erroneous behavior of any of the hosted applications or of the MCP to one system. If there are any timing problems between the applications hosted on the MCP, then only an application within the same system would be affected and the overall timing of the applications could be measured by only testing the applications from one system. With the limitation to one system, no application would be able to affect the operation of an application from another system.

e. Hyperthreading

Processors that use hyperthreading are not covered in this paper.

f. Use of any MCP with Only One Core Activated

Applicants intending to install an MCP but to only install software on one of the cores should ensure that any core without any software installed on it is deactivated and that any deactivated core does not interfere with the activated core or with the software hosted on it. They should also determine and set the configuration settings of the MCP to ensure that any mechanisms of the MCP that could interfere with the deterministic behavior of the hosted software are deactivated or mitigated. Objective MCP_Determinism_1 applies to such installations. Applicants for such installations should consider whether they need to meet objective MCP_Determinism_2 for their installation and apply it if they consider it necessary. Applicants for such installations should plan and conduct software verification of the software hosted by the one active core according to the applicable software guidance (e.g. AC 20-115C, version of ED12 / DO-178 that applies, EASA CRI, etc).

If an applicant for such an MCP configuration later decides to install software on a core that was previously deactivated, then the applicant should make a new application or, if the original system has already been certified, to consider such a change as a major change. Accordingly, the applicant should provide a new set of software and AEH documentation for the installation in which any additional core is activated, and that application should comply with this paper to the extent required for the DAL of the software hosted.

g. Exempted MCP Architectures

This paper does not apply to the following MCP architectures:

- Two core processors in which both cores host the same software and execute that same software in lock-step so that their outputs, based on identical input data, can be compared for use in a safety-critical application.
- Processors in which two identical or different activated cores are incorporated onto the same device, but the two activated cores are only linked by conventional data buses, and not by shared memory, shared cache, a 'coherency fabric / module / interconnect' or software or hardware hypervisor or any of the other features of MCPs that are described in this document. This category of MCPs includes any MCPs in which a core acting as a co-processor or a graphics processor is under the control of another core that executes software, provided that the cores are not connected by any of the MCP mechanisms described in this document. If the MCP contains any cores in addition to the two activated cores, it must be ensured that those other cores are deactivated and that they cannot cause any interference with the activated cores or with the software that executes upon the activated cores.

6. RATIONALE AND OBJECTIVES

a. Configuration Settings.

i. Rationale.

MCPs have many features and options that are configurable by setting the values of registers or pins. Inappropriate settings of some of these values could significantly change the way that the processor behaves so as to alter its outputs in an undesirable manner. Configuration settings may also be used to deactivate some of the features of MCPs that could produce non-deterministic behavior, or to deactivate features that should not be active during the normal operation of the MCP, such as undocumented debug, test, and performance monitoring features or uncontrolled dynamic features.

It is normal practice for applicants to determine the settings of configuration pins and software registers so that the features of the processor that will be used are activated and that any unused features are deactivated. However, MCPs contain more features than single core processors, some of the features of MCPs can cause interference between the software hosted on the two cores, and some of the dynamic features such as the ability to switch off a core while leaving one operating was not available until the introduction of MCPs. The certification authorities therefore consider that the control of configuration settings should be included in this paper.

Some MCPs may also have dynamic features that without intervention from outside the MCP or from the software executing on the cores, can dynamically alter the behavior of the processor whenever they detect that certain criteria programmed into the MCP are fulfilled. These features have generally been incorporated for use in installations that are not safety-critical, for reasons such as to save energy. These dynamic features may have the ability to dynamically

- Alter the frequency of operation of a core,
- Activate or deactivate a core, and
- Alter the allocation of memory within the device.

If the applicant does not control these dynamic features and maintain control over them during flight, they could alter the frequency of execution of a safety-critical application or shut it down so as to cause non-deterministic behavior of the hosted software.

ii. Objectives.
MCP_Determinism_1: The applicant has analyzed, determined and documented the configuration of the MCP settings for required, unused, and dynamic features that will be set either in hardware or in software during start-up and during operation and has verified that the use of those settings enables the MCP to execute the applications hosted on its cores in a deterministic manner with the software architecture and operating system(s) used in the intended installation. For undocumented features the applicant has contacted the MCP manufacturer to identify configuration settings used to control the undocumented features and has set those registers and pins to disable those features.
NOTE - such settings are typically part of the Hardware-Software Interface Data as per DO-254 /ED-80 paragraph 10.3.2.2.

MCP_Determinism_2: The applicant has planned, developed, documented, and verified a means that ensures that in the event of any safety critical configuration settings of the MCP being inadvertently altered an appropriate mitigation is implemented.

b. Processor Errata .

i. Rationale.

Most commercial processors including MCPs were not designed for safety applications. MCPs are so complex that they cannot be completely tested before production and will have residual design errors. MCP manufacturers rely on errata to notify their customers of errors uncovered in field use and provide suggested work-arounds by publishing an errata document. Even after years of wide commercial usage, design errors may still be uncovered and documented in a revision to the manufacturer's errata document. If the applicant does not have access to the complete list of errata for their selected MCP, or if those errata are not kept up to date by the manufacturer, then the applicant might not be aware of known faults in their selected MCP and would not, therefore, employ the necessary fault mitigation in their installation.

The certification authorities therefore consider that applicants need to be able to incorporate all necessary fault mitigation for the known faults of their MCP and also to be able to do so post-certification of the system.

ii. Objectives.

MCP_Determinism_3: The applicant has assessed the processor errata data provided by the manufacturer and has documented their processes for continuing to obtain errata from the manufacturer throughout the development and service life of the MCP installation and for resolving those problems in the same manner as any other reported problems. The applicant should provide an errata analysis that demonstrates the MCP maturity for their implementation (e.g. no new errata within the last year that would require a work around for the applicant's MCP implementation).

c. Software Hypervisors and MCP Hardware Hypervisor Features

i. Rationale.

Software Hypervisors: some MCP installations include a software hypervisor that is loaded onto the MCP and may be used for purposes such as to enable virtualization so that multiple virtual machines may be run on the same processor by abstracting the underlying processor cores, memory and devices. Such virtualization enables separate operating systems to be run on the separate cores of a multi-core processor, or even on a single core. Among the capabilities claimed for some software hypervisors are the abilities to build networks internal to the virtualized platform, to dynamically create, delete or migrate virtual machines to other CPU cores in real-time and to alter the frequency at which an application executes. A software hypervisor might not have been developed and verified according to DO-178B/ED-12B or DO-178C/ED-12C and might be a source of unintended functionality. Software hypervisors affect the execution of the software on both of the cores of the MCP and could cause non-deterministic behavior in the software hosted on the cores.

MCP Hardware based Hypervisors: some MCPs include hypervisors built into the MCP's hardware that can be used to provide a layer of supervision of the processes executing on the cores of the MCP, such as for controlling the overall settings of the MCP. These hypervisors are an extension of the already developed Supervisor function used in previous processor generations and are not usually fully documented by the processor manufacturer; so, they cannot be fully tested for compliance with any of the existing guidance material. These hypervisors could be a source of unintended functionality. These hypervisors might also affect the operation of the software hosted on both cores of an MCP and could cause non-deterministic behavior of the software.

ii. Objectives.
MCP_Determinism_4: The applicant has stated in their software/AEH plans or other deliverable documents whether or not they intend to use a software hypervisor or a MCP hardware based hypervisor in their MCP, and if they do, they have described for each part of the functionality of the hypervisor whether they intend to activate it, deactivate it or mitigate any undesirable behavior it may cause.
MCP_Determinism_5: If the applicant intends to use a software hypervisor, the applicant has stated in the software plans how they intend to show compliance of the software hypervisor with the certification authority's applicable guidance and has successfully conducted those activities that they planned. The applicant has also described in their plans any hardware features used by the software hypervisor and has verified those features.
MCP_Determinism_6: If the applicant intends to use an MCP hardware based hypervisor, the applicant has described in their AEH plans or other deliverable documents how they intend to verify the activated functionality of the MCP hardware based hypervisor and the applicant has verified all that functionality. The applicant has identified any parts of the MCP hardware based hypervisor that are deactivated and has verified that those parts have been deactivated.

d. MCP Interference Channels.

i. Rationale.

Applications running on different cores of a multi-core processor do not typically execute independently from each other because the cores are sharing resources. Even if there is no explicit data or control flow between these applications, a coupling exists on the platform level since they are implicitly sharing platform resources (e.g. multi-core shared cache memory, peripherals) and could result in interference between these applications. A platform property, that may cause interference between independent applications, is called an interference channel. The certification authorities are concerned that there may be software or hardware channels through which the MCP cores or the software hosted on those cores could interfere with each other, in addition to those channels specifically mentioned in this paper. Without mitigation being incorporated to deal with such interference, the certification authorities are concerned that non-deterministic behavior of the hosted software may occur.

ii. Objectives.

MCP_Determinism_7: The applicant has conducted a functional interference analysis to identify all the interference channels between the software hosted on the cores of the MCP and has designed, implemented and verified a means of mitigation for each of those interference channels.

e. Shared Memory and Cache.

i. Rationale.

Some MCPs have areas of memory or cache memory that are shared between the processing cores or that may be allocated so that they are shared.
The certification authorities are concerned that there have been documented cases in which the use of shared cache has resulted in the worst-case execution times (WCETs) of the software applications hosted on one core of an MCP increasing greatly due to repeated cache accesses by the processes hosted on the other core, leading to repeated cache misses.
The certification authorities are also concerned that the use of shared memory can lead to situations in which the software hosted on one of the cores may be locked out from accessing the shared memory locations. Such problems can cause applications to be unable to obtain the data they need at the time when they need it and could even result in an application halting due to memory access problems.

ii. Objectives.

MCP_Determinism_8: The applicant has stated in their software plans whether or not they intend to use shared memory (between the processing cores) and if they do, has described in those plans the means they intend to use to control access to shared memory locations and to prevent the disruptions to deterministic software execution caused by problems such as race conditions, data starvation, deadlocks or live-locks.

MCP_Determinism_9: If the applicant uses shared memory between the processing cores, the applicant has tested the means that they have designed to control the access to shared memory and has ensured that the implemented means provides uninterruptible access to the shared memory locations from either core of the MCP and prevents either core being locked out from accessing the shared memory.

MCP_Determinism_10: The applicant has stated in their software plans whether or not they intend to use shared cache between the processing cores, and if they do, has also described in their plans their strategy for managing and verifying cache usage.

MCP_Determinism_11: If the applicant uses shared cache between the processing cores, the applicant has conducted analyses and tests to determine the worst-case effects that the use of shared cache and memory can have on the execution of the specific software applications hosted on the two cores of the MCP, has described those effects to the certification authority, and has implemented and verified a means to mitigate the effects of using shared cache.

f. Planning and Verification of Resource Usage.

i. Rationale.

1) Interconnect Features.

Many MCPs include an interconnect feature (sometimes known as a 'coherency fabric' or 'coherency module') that controls the access of the processes executing on the two cores to memory, cache, core interconnects and the peripheral interfaces of the MCP. Processor manufacturers do not usually develop these features in accordance with any existing airborne guidance material and they do not usually provide documentation of the requirements or the code of these coherency features. The functionality of these features is not, therefore, usually fully verifiable. These features might not behave as expected and the features may include unintended functionality.

In some installations, the use of these features has been found to be a source of non-deterministic behavior of the processes executing on the two cores, with data arrival times varying in some cases and the access of the software hosted on a core to the peripherals of the device being blocked or interfered with.

Some interconnect features can cause transactions to either be lost or to be serviced in a different order from the order in which they were requested, or cause a service to be provided to the wrong requester all of which are non-deterministic behaviors. The certification authorities are concerned that the use of these features may result in incorrect transactions or in jitter in data arrival times.

2) Allocation of Functions, Scheduling and Analysis of Timing.
The cores of MCPs host software that executes in parallel and makes resource demands on the processing of the cores and on shared resources, such as memory, cache and peripherals. Access to these resources is often controlled via interconnect features that arbitrate between access requests, sometimes causing contention for access and imposing an overhead on the resource use of the software.
If the overall available resources of the MCP are exceeded by the combined resource demands of the separate software programs hosted on the MCP, the effects on the software hosted by the MCP may be unpredictable and the software may behave in a non-deterministic manner. The limit on the overall available resources of the MCP may be partly determined by the maximum capacity to provide deterministic transactions of any coherency mechanism used on the MCP. (See item 1above.)
The applicant should describe the measures they intend to take to avoid such problems, as well as any use of Configuration Files/Parameter Data Items to set and control the use of resources by the applications hosted on the MCP.

ii. Objectives.
MCP_Determinism_12: The applicant has described in their software/AEH plans or other deliverable documents how they intend to allocate, manage and measure the use of resources and the use of the interconnect by the applications hosted on the MCP and by other MCP peripherals so as to avoid contention for MCP resources and to prevent the capacity of the interconnect and the resources of the MCP from being exceeded.
MCP_Determinism_13: The applicant has allocated the usage of the MCP resources to the software applications hosted on the MCP and has verified that the total of the resource demands when all applications are executing in the worst-case situation does not exceed the total of the resources available.
MCP_Determinism_14: The applicant has determined the maximum capacity of any interconnect mechanism of their MCP to sustain transactions in a deterministic manner and has verified that the demands made on that mechanism by the software hosted on the MCP or by any peripherals of the MCP do not exceed its maximum capacity during any phase of operation of the system.
MCP_Determinism_15: The applicant has identified the non-deterministic effects that any coherency mechanism of their MCP could cause, such as

excessive jitter in data arrival times, or transactions being lost or serviced in an undesirable order, and has designed, implemented and verified means to deactivate the features concerned or to mitigate the non-deterministic effects.

g. Software Planning and Development Processes.

i. Rationale.

DO-178B/ED-12B and DO-178C/ED-12C only address software that is installed on a single-core processor. It does not address the development or verification of software that executes in parallel in real-time on the separate cores of an MCP. MCP features and the software architecture of the MCP may introduce situations in which the applications executing on the separate cores could cause interference with each other's execution. Developing and verifying the software installed on each of the MCP cores as separate software programs would not address these MCP-specific problems and would therefore not be sufficient to ensure that the software would behave according to its requirements and in a safe and deterministic manner when all the software applications were executing in parallel.

There is no existing guidance material that specifies how DO-178B/ED-12B and DO-178C/ED-12C development and verification might be adapted to cover the software installed on MCPs. The certification authorities are therefore concerned that if applicants attempt to conduct verification of software hosted on MCPs, there is insufficient existing guidance for them to ensure that the applications hosted on two cores will fulfill their requirements when all the software is executing.

Three types of software architecture are currently used with MCPs. These are Symmetric Multi-processing (SMP), Bound Multi-processing (BMP) and Asymmetric Multi-processing (AMP), for which the definitions may be found above.

In existing certified systems, each software application is statically allocated to execute on a designated core within designated memory boundaries for their code and variables as a result of software integration. The use of dynamic software features such as those of object-oriented languages for dynamic tasking and memory management has been deliberately restricted so as to provide a predictable and deterministic environment for the execution of safety-critical software.

With SMP, processing threads can be dynamically reallocated by the processor or its single operating system during operation.

The certification authorities are concerned that neither the existing guidance nor the existing industry practices for deterministic, safety-critical software cover the development, integration or verification of threads that are controlled in this dynamic manner or cover the OS that controls the threads.

Applicants for the use of dynamic software architectures should demonstrate how they intend to adapt the existing guidance and their software processes for use with their selected OS and with the applications to be hosted by their selected dynamic architecture, and how they intend to demonstrate that the hosted software behaves deterministically.

ii. Objectives.

MCP_Software_1: In the software plans, the applicant has identified their MCP software architecture and operating system(s) and has provided details of how they will develop and verify all the software loaded onto the MCP so as to ensure that the software applications hosted by the MCP execute deterministically including details of any special methods or tools that are necessary due to the use of an MCP or the selected MCP architecture.

h. Software Verification.

i. a) Rationale.

DO-178B/ED-12B and DO-178C/ED-12C only describe processes to be used to verify software hosted on single core processors. Since there is currently no existing guidance on how to conduct software verification on an MCP and the use of MCPs in safety-critical systems is new, there is no existing standard industry-wide practice regarding how to conduct verification for software hosted on an MCP.

Because of the lack of industry experience with MCPs, the certification authorities are concerned that when applicants attempt to conduct software verification using only the existing guidance, their approach to verification may not provide enough assurance that all the software would comply with its requirements when executing in parallel with the mechanisms of the MCP permitting interference between the software hosted on the two cores. Additional guidance is necessary regarding the verification of software installed on MCPs.

One area in which there is existing guidance and standard industry practice regarding the integration and verification of hardware platforms, operating systems and applications hosted upon them is in the field of IMA systems. In such systems, the OS and any other hardware / software interface is integrated and tested with the host hardware first, then each application is individually integrated and tested with the OS and the hardware. Finally, the correct operation of the entire system is verified. Reference DO-297/ED-124 section 3.1.3 d. 1) and 2).

The certification authorities consider that a similar approach of integrating and then verifying components gradually would be an effective approach to the verification of software on an MCP and since it is an approach that is already widely in use, it would not impose any additional burden on industry. The main differences in using this approach with an MCP would be that there are operating systems or hardware / software interfaces on two cores to integrate and test and then applications to integrate on two cores instead of one. This approach could then be extended by testing the robustness of the software and of the MCP mechanisms when all the software is executing.

If such an organized and gradual traditional approach to software development, integration and verification is not used, it will be difficult to attribute the errors that occur during verification to their root causes and to ensure that all the software complies with its requirements and executes in a deterministic manner. Many of the problems that may occur during testing of software hosted on MCPs may be due to the internal mechanisms of the MCP itself. Therefore it important that testing is conducted with the software under test installed on the target MCP as in the final intended hardware environment.

ii. Objectives.

MCP_Software_2: The applicant has described in their Software Verification Plan (SVP) the environment to be used for each software test activity, and for any testing that will not be conducted using the target MCP, the applicant has described the environment that they intend to use, their rationale for using a different test environment and why they consider that test environment to be sufficiently representative of the target MCP.

MCP_Software_3: The applicant has verified that each operating system and/or software interface with the MCP hardware when installed on the MCP target complies with applicable objectives in DO-178B/ED-12B or DO-178C/ED-12C (e.g. DO-178C reference A-5 numbers 1 through 9, A-6 numbers 1 through 5) .

MCP_Software_4: The applicant has verified that each individual software application that is hosted on the MCP complies with the applicable objectives in DO-178B/ED-12B or DO-178C/ED-12C (e.g. DO-178C reference A-5 numbers 1 through 9, A-6 numbers 1 through 5) when all the applications hosted on the MCP are executing in the intended final configuration of the processor and its hosted software.

MCP_Software_5: The applicant has verified that the data and control coupling between all software components hosted on the MCP has been exercised during software requirement-based testing including exercising any implicit (e.g. through interconnect features) or explicit interfaces between the applications via shared memory and any mechanisms to control the access to shared memory, and that the data and control coupling is correct.

MCP_Software_6: The applicant has conducted robustness testing of the interfaces and the features of the MCP both when software applications are executing individually and when all the software hosted on the MCP is executing, and has verified the compliance of all the hosted software with the applicable objectives in DO-178B/ED-12B or DO-178C/ED-12C (e.g. DO-178C reference A-6 numbers 1 through 5) and with the resource allocation to each application under these conditions.

i. Discovery of Additional Features or Problems.

i. Rationale.

There may be other features of MCPs of which the certification authorities are not currently aware, or which may be introduced in new MCPs and are not already covered by any section of this paper. These features might introduce additional channels for interference between the MCP cores or might introduce some other undesirable non-deterministic behavior.

During the development of the AEH or the software of the MCP installation, problems or additional features of an MCP may be discovered that make invalid some of the original assumptions or analyses concerning the feasibility of using the MCP. The certification authorities are concerned that any such additional problems could affect the behavior and determinism of the hosted software so the certification authorities request that applicants inform them of any such problems.

ii. Objectives.

MCP_Determinism_16: The applicant has identified any features of the MCP not specifically mentioned in this paper that could cause non-deterministic behavior of the software hosted on the MCP, has designed, implemented and tested means to deactivate those features or mitigate their effects and has informed the certification authority of the features and the means of deactivation or mitigation in the applicable AEH or software plans or subsequent documentation if the problems are discovered at a later stage.

j. Error Detection and Handling and Safety Nets.

i. Rationale.

As well as the types of errors and failures normally detected and handled in a system that incorporates a single core processor (e.g. errata), additional types of errors and failures may need to be detected and handled in an MCP environment due to problems caused by the features of MCPs and due to the additional complexity of executing two software programs on two cores in parallel and real-time.

There are MCP built-in features (e.g. coherency module) for which the manufacturer does not provide full documentation; so, those features cannot be thoroughly tested by the applicant and cannot be verified for compliance with the existing guidance material.

The certification authorities are concerned that the features of an MCP may therefore contain unintended functionality that may cause errors and produce behavior that is not normally seen with single core processors. Therefore, additional mechanisms would need to be developed and verified to detect and handle these specific errors associated with these features.

These problems may not be specific to MCPs, however, there is no existing guidance that describes the use of Safety Nets with MCPs, which have different and more complex mechanisms than conventional single core processors and which may therefore have different and more complex failure characteristics to detect and mitigate.

ii. Objectives.

MCP_Error_Handling_1: The applicant has identified the various types of errors or failures that may occur within the MCP or the software hosted upon it and has planned, designed, implemented and verified means, including a 'safety net' that is external to the MCP, by which to detect and handle those errors or failures in a fail-safe manner that contains the effects of any errors or failures within the system in which the MCP is installed.

MCP_Error_Handling_2: If part of the safety-critical functionality hosted by the applicant's MCP must continue to be available even after errors or failures are detected in the MCP or its hosted software, the applicant has designed, implemented and verified a means to provide that functionality that is external to the MCP.

NOTE – this separate means is for use in case resetting the MCP does not restore the required functionality.

k. Summary of MCP paper Objectives with their DAL Applicability

MCP OBJECTIVES – the first of the DAL columns shows the objectives applicable when the highest DAL of the hosted software applications is DAL A or B, and the second column shows which objectives apply if the highest DAL of the hosted software is DAL C.	DAL A or B	DAL C
MCP_Determinism_1: The applicant has analyzed, determined and documented the configuration of the MCP settings for required, unused, and dynamic features that will be set either in hardware or in software during start-up and during operation and has verified that the use of those settings enables the MCP to execute the applications hosted on its cores in a deterministic manner with the software architecture and operating system(s) used in the intended installation. For undocumented features the applicant has contacted the MCP manufacturer to identify configuration settings used to control the undocumented features and has set those registers and pins to disable those features.	Y	Y
MCP_Determinism_2: The applicant has planned, developed, documented, and verified a means that ensures that in the event of critical configuration settings of the MCP being inadvertently altered an appropriate mitigation is implemented.	Y	Y
MCP_Determinism_3: The applicant has assessed the processor errata data provided by the manufacturer and has documented their processes for continuing to obtain errata from the manufacturer throughout the development and service life of the MCP installation and for resolving those problems in the same manner as any other reported problems. The applicant should provide an errata analysis that demonstrates the MCP maturity for their implementation (e.g. no new errata within the last year that would require a work around for the applicant's MCP implementation).	Y	Y
MCP_Determinism_4: The applicant has stated in their software/AEH plans or other deliverable documents whether or not they intend to use a software hypervisor or an MCP hardware based MCP hypervisor in their MCP, and if they do, they have described for each part of the functionality of the hypervisor whether they intend to activate it, deactivate it or mitigate any undesirable behavior it may cause.	Y	

MCP_Determinism_5: If the applicant intends to use a software hypervisor, the applicant has stated in the software plans how they intend to show compliance of the software hypervisor with AC 20-115C and has successfully conducted those activities that they planned. The applicant has also described in their plans any hardware features used by the software hypervisor and has verified those features.	Y	Y
MCP_Determinism_6: If the applicant intends to use an MCP hardware based hypervisor, the applicant has described in their AEH plans or other deliverable documents how they intend to verify the activated functionality of the MCP hardware based hypervisor and the applicant has verified all that functionality. The applicant has identified any parts of the MCP hardware based hypervisor that are deactivated and has verified that those parts have been deactivated.	Y	
MCP_Determinism_7: The applicant has conducted a functional interference analysis to identify all the interference channels between the software hosted on the cores of the MCP and has designed, implemented and verified a means of mitigation for each of those interference channels.	Y	
MCP_Determinism_8: The applicant has stated in their software plans whether or not they intend to use shared memory (between the processing cores) and if they do, has described in those plans the means they intend to use to control access to shared memory locations and to prevent the disruptions to deterministic software execution caused by problems such as race conditions, data starvation, deadlocks or live-locks.	Y	Y
MCP_Determinism_9: If the applicant uses shared memory between the processing cores, the applicant has tested the means that they have designed to control the access to shared memory and has ensured that the implemented means provides uninterruptible access to the shared memory locations from either core of the MCP and prevents either core being locked out from accessing the shared memory.	Y	Y
MCP_Determinism_10: The applicant has stated in their software plans whether or not they intend to use shared cache between the processing cores, and if they do, has also described in their plans their strategy for managing and verifying cache usage.	Y	Y
MCP_Determinism_11: If the applicant uses shared cache between the processing cores, the applicant has conducted analyses and tests to determine the worst-case effects that the use of shared cache can have on the execution of the specific software applications hosted on the two cores of the MCP, has described those effects to the certification authority and has implemented and verified a means to mitigate the effects of using shared cache.	Y	Y

	MCP_Determinism_12: The applicant has described in their software/AEH plans or other deliverable documents how they intend to allocate, manage and measure the use of resources and the use of the interconnect by the applications hosted on the MCP and by other MCP peripherals so as to avoid contention for MCP resources and to prevent the capacity of the interconnect and the resources of the MCP from being exceeded.	Y	Y
	MCP_Determinism_13: The applicant has allocated the usage of the MCP resources to the software applications hosted on the MCP and has verified that the total of the resource demands when all applications are executing in the worst-case situation does not exceed the total of the resources available.	Y	
	MCP_Determinism_14: The applicant has determined the maximum capacity of any interconnect mechanism of their MCP to sustain transactions in a deterministic manner and has verified that the demands made on that mechanism by the software hosted on the MCP or by any peripherals of the MCP do not exceed its maximum capacity during any phase of operation of the system.	Y	Y
	MCP_Determinism_15: The applicant has identified the non-deterministic effects that any coherency mechanism of their MCP could cause, such as excessive jitter in data arrival times, or transactions being lost or serviced in an undesirable order, and has designed, implemented and verified means to deactivate the features concerned or to mitigate the non-deterministic effects.	Y	
	MCP_Determinism_16: The applicant has identified any features of the MCP not specifically mentioned in this paper that could cause non-deterministic behavior of the software hosted on the MCP, has designed, implemented and tested means to deactivate those features or mitigate their effects and has informed the certification authority of the features and the means of deactivation or mitigation in the applicable AEH or software plans or subsequent documentation if the problems are discovered at a later stage.	Y	
	MCP_Software_1: In their software plans, the applicant has identified their MCP software architecture and operating system(s) and has provided details of how they will develop and verify all the software loaded onto the MCP so as to ensure that the software applications hosted by the MCP execute deterministically, including details of any special methods or tools that are necessary due to the use of an MCP or the selected MCP architecture.	Y	Y

MCP_Software_2: The applicant has described to the certification authority in their SVP the environment to be used for each software test activity, and for any testing that will not be conducted using the target MCP, the applicant has described the environment that they intend to use, their rationale for using a different test environment and why they consider that test environment to be sufficiently representative of the target MCP.	Y	Y
MCP_Software_3: The applicant has verified that each operating system and/or software interface with the MCP hardware when installed on the target MCP complies with the applicable objectives in DO-178B/ED-12B or DO-178C/ED-12C (e.g. DO-178C reference A-5 numbers 1 through 9, A-6 numbers 1 through 5) .	Y	
MCP_Software_4: The applicant has verified that each individual software application that is hosted on the MCP complies with the applicable objectives in DO-178B/ED-12B or DO-178C/ED-12C (e.g. DO-178C reference A-5 numbers 1 through 9, A-6 numbers 1 through 5) when all the applications hosted on the MCP are executing in the intended final configuration of the processor and its hosted software.	Y	Y
MCP_Software_5: The applicant has verified that the data and control coupling between all the software components hosted on the MCP has been exercised during software requirement-based testing, including exercising any implicit (e.g. through interconnect features) or explicit interfaces between the applications via shared memory and any mechanisms to control the access to shared memory, and that the data and control coupling is correct.	Y	Y
MCP_Software_6: The applicant has conducted robustness testing of the interfaces and the features of the MCP, both when software applications are executing individually and when all the software hosted on the MCP is executing, and has verified the compliance of all the hosted software with the applicable objectives in DO-178B/ED-12B or DO-178C/ED-12C (e.g. DO-178C reference A-6 numbers 1 through 5) and with the resource allocation to each application under these conditions.	Y	Y
MCP_Error_Handling_1: The applicant has identified the various types of errors or failures that may occur within the MCP or the software hosted upon it and has planned, designed, implemented and verified means, including a 'safety net' that is external to the MCP, by which to detect and handle those errors or failures in a fail-safe manner that contains the effects of any errors or failures within the system in which the MCP is installed.	Y	Y

MCP_Error_Handling_2: If part of the safety-critical functionality hosted by the applicant's MCP must continue to be available even after errors or failures are detected in the MCP or its hosted software, the applicant has designed, implemented and verified a means to provide that functionality that is external to the MCP.	Y	

l. Conclusion

Section 5 of this paper presented the certification authorities position for MCPs with two active cores by describing the rationale for why the topics of concern should be addressed and by identifying the objectives the applicant should address for each topic. Appendix A suggests activities that the certification authorities deem sufficient for the each topic of concern. These activities are not mandatory.

This paper was limited to MCPs with two active cores and single system implementations. This paper may be extended in future to address MCPs with more than two active cores and MCP IMA implementations.

Appendix A: Multi-Core Processors, Suggested Activities

1. Configuration Settings.

The applicant should –

a. conduct an analysis of all the possible configuration settings of the proposed MCP, including the settings to activate or deactivate any required, unused, dynamic and undocumented features of the MCP,

b. determine the selected values of the configuration settings that they intend to use during each operational phase (e.g. initial power up, power on reset, BIT, exception processing)so as to provide the required functionality of the device, while disabling any unused, undocumented and non-deterministic features (such as those described in this document, or any others that may be discovered),

c. capture any settings of hardware configuration pins as hardware requirements to ensure that they are correctly implemented, and capture any software register settings as software high-level or low-level requirements,

d. set the values of the software-programmable registers to the selected values during the start-up of the software of the two cores,

e. design and implement a means of mitigation for use if any of the software-programmable register configuration settings are inadvertently altered, such that the mitigation enables the software hosted by the MCP to continue to behave in a safe and deterministic manner. Possible ways to do this could involve executing the activities of either item i. or item ii. below, and applying item iii.:
   i. testing the values of the registers that are critical for safe and deterministic behavior of the hosted software to ensure that they have not been altered, and then resetting the values of any inadvertently-changed registers,
   ii. regularly overwriting the values of all the registers or at least of the registers that are critical for safe and deterministic behavior of the hosted software,
   iii. executing any such mitigation functions at an interval determined by the applicant so as to minimize the time window during which the MCP could behave in an undesirable manner due to any of these settings having been unintentionally altered by the software or by external influences such as single event upsets

f. document their configuration design, analysis, settings and test results.

2. <u>Processor Errata.</u>

The applicant should

a. provide evidence to show:
   - How the component manufacturer captures and maintains the list of errata and publishes it.
   - The maturity of the device, e.g. time in service since release, the number of different applications that have used the MCP, rate of occurrence of new errata, etc.

b. assess all the errata from the component manufacturer for any potential adverse safety effects on the system. This assessment should comprise:
   - Justification to show which of the errata are applicable to the specific application of the device,
   - Justification to show which of the errata are not applicable to the specific application of the device,
   - A description of the mitigation implemented for each of the applicable errata,
   - Evidence that the implementations of errata mitigations are covered by relevant requirements and design data and verified.

c. document their own past experience of usage of the component and the experience they gained as part of the current development, along with any other additional recommendations (e.g. errata workarounds) that should be implemented in order to use the component.

d. analyze any errata discovered either prior to certification or post certification for their functional, operational and safety impacts and resolve those problems in the same manner as any other reported problems.

3. <u>Software Hypervisors and MCP Hardware Based Hypervisors.</u>

The applicant should -

a. state in their initial certification briefing and in their software or AEH plans whether or not their installation of their selected MCP will involve the use of a software hypervisor or an MCP hardware based hypervisor.

b. provide a rationale for its use,

c. describe the functionality of the software hypervisor or MCP hardware based hypervisor that would be used in the proposed system,

d. state how they intend to verify the functionality of the software hypervisor or MCP hardware hypervisor feature,

e. state which functionality of the software hypervisor or MCP hardware based hypervisor is not used in the proposed installation,

f. describe how they will ensure that all of the un-used functionality of the software hypervisor or MCP hardware based hypervisor is deactivated and remains deactivated during operation of the MCP,
g. state whether any software hypervisor was developed and verified for compliance with AC 20-115C and if not, state how they intend to show compliance of the feature with the AC 20-115C at the same DAL as the highest DAL application hosted by the MCP, or a higher DAL, and then carry out those activities,
h. analyze the features of any software hypervisor or MCP hardware based hypervisor and determine, which of those features could cause interference between any of the software applications hosted on the MCP or could result in any other non-deterministic behavior,
i. design, implement and verify a means of mitigation for any features that they identify as being capable of causing interference between the applications hosted on the cores or any other non-deterministic behavior,

4. <u>MCP Interference Channels.</u>

The applicant should -

a. conduct an interference analysis in order to identify all the channels by which the software programs executing on the two cores could interfere with each other via the internal mechanisms of the MCP or through any of the software hosted in it, including the kinds of features of MCPs described in this paper that are not present in single core processors.
b. for each of the interference channels identified, design a means to deactivate the interference channel or to mitigate the effects of the interference between the software hosted on the separate cores. This interference analysis should be available for review during audits.

5. <u>Shared Memory and Cache.</u>

The applicant should -

a. develop and implement means to control access to the shared memory areas by the software hosted on the cores of an MCP. The resulting implementation should prevent situations in which an access to the shared memory by the software hosted on one core does not cause disruption to the execution of the software hosted by another core due to such problems as race conditions, data starvation, deadlocks, or live-locks.
b. analyze the means that are used to communicate between the cores via shared memory. This supporting analysis should be documented. The resulting implementation should be documented in the applicable software and AEH requirements and design data.
c. describe in their software plans their strategy for managing cache memory and state whether or not they intend to use shared cache.

d. if shared cache is going to be used, state in the software plans how they intend to mitigate any interference this may cause between the applications executing on the two cores of the MCP.
e. if shared cache is used, conduct analyses and tests to determine for the software hosted on each core the worst extent of the effects due to the use of shared cache in terms of aspects such as data corruption, scheduling and the WCET due to accesses to cache from the other core. The applicant should allow for these effects in the allocation of processing time to the processes hosted on the two cores.
f. document these analyses and tests and their results for the certification authority review and provide preliminary results as part of the justification that a proposed MCP installation is feasible and can be fully verified.

6. Planning and Verification of Resource Usage.

If the applicant intends to use an MCP with an interconnect/coherency mechanism, the applicant should -

a. conduct analyses and tests to determine the maximum capacity of the mechanism to sustain transactions in a deterministic manner. The applicant should verify that the demands made on that mechanism by the software hosted on the MCP will not exceed that maximum capacity during any phase of operation of the system,
b. conduct analyses and tests to determine the worst-case extent of the perturbations that can be caused by the interconnect/coherency features that they use on their selected MCP and design mitigating requirements and actions to handle any such perturbations,
c. conduct analyses and tests to determine whether non-deterministic behavior can be caused by the interconnect mechanism, such as transactions being lost or being serviced in an undesirable order or causing excessive jitter in data arrival times, and implement and test means to prevent such non-deterministic behavior from occurring,
d. deactivate (if possible) any parts of coherency features that are unused,
e. conduct analyses and testing to verify that their planned mitigations for effects of interconnect/coherency features have the intended effects and have been implemented correctly,
f. describe their mitigating actions planned in the PSAC and PHAC as part of the justification that the proposed MCP installation is feasible.
g. conduct a detailed analysis into the timing of processes on the two separate cores and of all the resource accesses of the two programs to be installed on the MCP. This analysis should include the relations between the timing of the processes hosted on the two cores and the timing of all resource accesses and interactions between the software hosted on each core and the memory or peripherals of the MCP. This analysis should:

- ensure that in all deliberate interactions (such as data transfers) between the two programs, the transmitting program will have time to gather the data before transferring it and the receiving program will have time to collect the data, allowing for any jitter in the timing of the execution of the programs or in the gathering of the data by the transmitting program,
- avoid situations in which the resource demands of the programs hosted on the two cores clash with each other and could cause overloads of the internal mechanisms of the MCP, such as any interconnect/coherency mechanism,
- avoid situations in which race conditions, data starvation or deadlocks could occur.

h. These analyses and test results should be documented and made available for review during audits. Preliminary data and test results should be provided during the initial certification briefing as part of the justification of the feasibility of the proposed MCP installation.

7. <u>Software Planning and Development Processes.</u>

The applicant should

a. identify in their software plans the type of software architecture and the types of operating system (if any) that they intend to use.

b. provide in their software plans details of how they will develop and verify all the software hosted on the MCP so as to show that it complies with AC 20-115C and behaves in a deterministic manner.

c. include in their plans details of how they intend to comply with all applicable software objectives and the objectives of this document.

d. Software Development Plan (SDP) should provide details of the methods and tools to be used to develop all the software that will execute on the two processor cores within the chosen software architecture. The SDP or the Requirement, Design and Coding Standards should include any necessary requirements and constraints on the allocation of functions to the cores or the allocation of resource accesses, and on the ways in which requirements, software designs and software code will be specified so as to execute on the separate cores of the target MCP.

e. Software Verification Plan (SVP) should include the planning of all the verification activities detailed in the Software Verification section, along with any additional requirements and activities that the applicant considers to be necessary or desirable.

8. <u>Software Verification.</u>

a. Since many of the problems related to the use of MCPs are caused by the internal features and architectures of the MCPs themselves, it is vital that the software verification is conducted on the target processor so as to determine the effects of the processor features on the execution of the programs. If the applicant finds it necessary to conduct any of the software verification activities in an environment that does not use the target hardware of the MCP, then they should justify this in their SVP.

b. In the case of an MCP, the respective OS(s) and/or any hardware/software driver interfaces should be integrated and verified with their separate cores, then each of the applications should be developed separately and then integrated and verified separately with the OS, as is normal industry practice for conventional software verification and for IMA systems. Finally the whole set of applications should be integrated and verified. The interfaces of the MCP and its internal mechanisms should be put under stress and robustly tested when all the software is installed and executing.

c. Conducting the verification in this manner, is similar to the existing standard practices used with an IMA. These suggested activities include very few activities in addition to those usually performed for an IMA system. This method also fits in perfectly with the allocation and measurement of resource usage called for in the determinism objectives of this document (which are also similar to existing practices used for IMAs).

d. Data and control coupling analyses should be extended by the applicant in both their planning (in the SVP) and in their verification to cover all the software components installed on the MCP cores (such as the OS(s) and the applications) and all the interactions between those software components. DO-178C/ED-12C states that data and control coupling should be exercised (and not merely confirmed) during requirement-based testing. Since the data and control coupling between the programs executing on the separate cores of an MCP is very important, the data and control coupling should actually be exercised, even in DO-178B/ED-12B installations that involve MCPs. Specific test cases and procedures may have to be developed in order to stimulate, exercise and verify the data and control coupling between the two software programs hosted on the two cores, particularly any interfaces via shared memory.

e. Applicant should state in their SVP how they intend to determine the worst-case execution time of the software applications executing on the separate cores of the MCP when all the software is installed and executing in parallel and the applicant should document the results for review by the certification authority.

9. Discovery of Additional Features or Problems.

The applicant should –

a. analyze the manufacturer's data for their selected MCP and identify any features of the MCP and document these features in their AEH plans that could cause non-deterministic behavior of the software hosted on the MCP and are not present in single core processors and are not covered in this paper.

b. inform the certification authority during the development or verification of a system that use an MCP if any of the following features, behaviors, information, or problems were discovered and how they will be addressed or mitigated:
   - additional features or behaviors of the proposed MCP that were not previously dealt with and not previously described to the certification authority
   - information they initially provided to the certification authority was incorrect or incomplete,
   - unexpected problems with the MCP or the software hosted on it that are likely to affect the ability of the MCP to provide safe and deterministic execution of its hosted software

10. <u>Error Detection and Handling, and Safety Nets.</u>

The applicant should -

a. analyze the types of failures or disturbances that could occur, for example:

   - in the MCP itself,

   - in the software executing on either of the cores,

   - due to interactions between the software hosted on the two cores,

   - due to any of the mechanisms of the device (such as those mentioned above)

   and design ways of detecting and safely handling all the identified failures, errors or disturbances. The applicant should conduct analyses of the kinds of undesirable effects that could occur due to the features of an MCP and design means to detect and mitigate those effects. Some of these effects should be readily detectable and able to be handled by the OSs of the cores, such as increases in the worst-case execution time of a process, however, other effects may be more subtle and means should be designed to detect the symptoms they produce and to mitigate their effects. The software planning should include analysis of whether or not any of the partitions of the processes hosted on the cores should be shut down in the event of errors being detected within those partitions, which errors can be tolerated and which should result in a hard or soft reset of one or both of the cores.

b. in either the software or AEH plans or other deliverable document the requirements, analyses and the measures taken to detect and mitigate errors at all levels of the MCP and its software.

c. analyze whether any safety-critical application could be halted, aborted or made non-functional (e.g. if needed input data was no longer produced by an aborted application) due to the loss of another application or the loss of another core on the same processor or due to any erroneous behavior of the MCP. If such a situation could occur, the applicant should identify what could cause this to happen, the outcome it would produce and how this would be handled in their MCP implementation.

d. For systems using MCPs that incorporate features for which full documentation is not available and the features cannot, therefore, be tested as to show compliance with the existing software and AEH guidance material, design and implement a 'safety net' within the system to detect errors or disturbances and handle them in a safe manner. A 'coherency fabric/module' is one of these features that cannot usually be fully tested due to the lack of complete MCP manufacturer's data.

e. There may be systems in which some of the safety-critical functionality allocated to an MCP and its hosted software must continue to be available even after the detection and handling of errors in the MCP or its hosted software. For such systems, resetting the cores and restarting the software might be sufficient to restore the required functionality. However, for such systems, the applicant should consider, analyze and implement measures to continue to provide the required functionality in the event that restarting the software a limited number of times fails to restore that required functionality or the system's response timing could not tolerate the time required to restart the software. Those measures should include the provision of a separate and different device (within the same system) that can be verified for compliance with the existing software and AEH guidance material.

   As an example, an DO-254 compliant FPGA or ASIC could be designed and installed for use when the safety net of the MCP has attempted to restart the MCP a limited number of times and the correct functionality of the system has not been restored. Such a device would be designed and verified to provide the minimum subset of the functionality allocated to the MCP and its hosted software that is necessary for continued safe flight and landing.

f. When a safety net is implemented to detect errors, handle them, to monitor the number of attempted resets of the MCP and to switch the system to use the backup device, it should be developed and verified to at least the same development assurance level (DAL) as that of the functionality allocated to the MCP.

In the event that the system reverts to the backup functionality of the kind of separate device suggested above, which may involve the functionality being degraded, then the certification authorities suggest the crew should be informed that such a reversion has taken place.